JS Level 0

Введение

Введение

Сегодняшняя наша лекция будет посвящена фундаментальным вещам, которые позволят вам понять, что из себя представляет современный Web и как с ним работать (если вы хотите стать разработчиком).

HTML5

HTML5

Чтобы понимать используемые в настоящее время термины HTML, нужно немного знать историю развития HTML. О чём мы с вами и поговорим в этом разделе.

Изначально HTML появился в 1990 годах и был предназначен для публикации научных документов (статей). Развивался в рамках организации CERN (Европейская организация по ядерным исследованиям), затем IETF (Internet Engineering Task Force) – Инженерный Совет Интернета.

HTML5

В 1995 году была создана организация W3C (World Wide Web Consortium) – ответственная за разработку Web стандартов, т.е. документов, описывающих, как должны работать технологии, которые используются в современнов Web'e.

В рамках этой организации в HTML добавлялись новые возможности, описываемые в спецификациях и им присваивался соответствующий номер, например:

- 3.2 (1995 год)
- 4.0 (1997 год)
- 4.01 (1999 год)

HTML5

Затем с 2000 года W3C сфокусировалась на разработке более строгого стандарта XHTML (который не прижился).

Разногласия с позициями W3C привели к созданию группы WHATWG (Web Hypertext Application Technology Working Group – Рабочая Группа по Вебу, Гипертексту, Приложениям и Технологиям).

Именно WHATWG сфокусировалась на том, чтобы дальше развивать HTML, который и получил название HTML5.

HTML5

Что получилось в итоге:

- WHATWG создал так называемый Living Standard (живой стандарт) без версий, в который постоянно вносятся улучшения, а создатели браузеров постоянно реализуют эти улучшения
- W3C периодически публиковала нумерованные версии спецификаций (т.е. описаний): например 5.2, 5.3
- Начиная с 2019 года W3C и WHATWG работают вместе на площадке WHATWG над развитием спецификаций HTML и DOM

HTML5

Что из этого нужно запомнить:

1. HTML постоянно развивается, поэтому если вы видите статьи более чем годичной давности, не стоит их читать (тем более, если там написано HTML4 и что-то подобное)
2. HTML5 – это обозначение живого стандарта, который можно найти на https://html.spec.whatwg.org/multipage/

Особенности HTML

HTML5

Если мы начнём читать спецификацию, то увидим следующую фразу:

Features have thus arisen from many sources, and have not always been designed in especially consistent ways. Furthermore, because of the unique characteristics of the web, implementation bugs have often become de-facto, and now de-jure, standards, as content is often unintentionally written in ways that rely on them before they can be fixed.

Перевод ключевой идеи: добавляемые функции не всегда были спроектированы "правильно" и последовательно (т.е. логично). Кроме того, реализации этих функций с ошибками (в браузерах) становились настолько распространёнными, что вместо того, чтобы исправлять их, было принято решение сделать их стандартом (т.е. вместо того, чтобы исправить ошибку, было принято описать её как стандартное поведение).

HTML5

Поэтому к некоторым "нелогичностям" и недоработкам нужно будет привыкнуть (как и ко многому в мире IT).

HTML5

Кроме того, если вы делали предыдущее ДЗ с видео, могли заметить, что наличие атрибута autoplay не приводит к автоматическому проигрыванию видео (хотя атрибут сам по себе об этом говорит).

В спецификации по этому поводу сказано:

Note

Authors are urged to use the `autoplay` attribute rather than using script to trigger automatic playback, as this allows the user to override the automatic playback when it is not desired, e.g. when using a screen reader. Authors are also encouraged to consider not using the automatic playback behavior at all, and instead to let the user agent wait for the user to start playback explicitly.

Перевод ключевой идеи: Авторам также рекомендуется вообще не использовать автоматическое воспроизведение, а вместо этого позволить пользовательскому агенту ждать, пока пользователь явно не начнет воспроизведение.

HTML5

Что на самом деле: разработчики браузеров стремятся оградить своих пользователей от "назойливого контента" – всплывающих окон, автоматически воспроизводящихся аудио и видео, уведомлений, включения микрофона/камеры, определения геолокации, выбора или сохранения файлов на компьютере пользователя или интенсивного использования ресурсов.

Поэтому, новые версии браузеров всё больше ограничивают веб-страницы в их возможностях: т.е. если вы даже напишите атрибут autoplay, то видео может не начать воспроизводиться в режиме со звуком, чтобы не мешать пользователю. А вот без звука (атрибут muted) – вполне.

HTML5

Т.е. вот такой код в браузере Chrome не будет автоматически воспроизводить видео:

```
<video src="https://alif-skills.pro/media/spring.webm" autoplay controls></video>
```

А вот такой будет:

```
<video src="https://alif-skills.pro/media/spring.webm" autoplay muted controls></video>
```

И с каждым годом/месяцем таких ограничений становится всё больше, поскольку Web – это среда растущих возможностей: там можно платить, передавать аудио/видео с камеры/микрофона, геолокацию и т.д.

HTML5

Что всё это значит для нас:

1. Web не стоит на месте, если вы хотите работать в этой сфере, то нужно быть готовым(ой) к тому, что придётся всё время следить за новостями, узнавать новое и учиться/переучиваться
2. То, что работало сегодня, может перестать работать в новых браузерах через месяц/год
3. То, что работает в одном браузере, может не работать или работать по-другому в другом браузере

Элемент HTML

Перейдём теперь к обсуждению ключевых элементов HTML и особенностей работы с ними.

Элемент html

html

Первый элемент, который у нас есть – это html. Этот элемент определяет корень html-документа.

Q: что значит корень?

A: HTML-документы строятся по принципу иерархии: у каждого элемента (кроме корневого) может быть ровно один родительский (тот, в который он вложен) и множество дочерних (которые вложены в него).

html

```
<!DOCTYPE html>
<html lang="en">
<head>
</head>
<body>
</body>
</html>
```

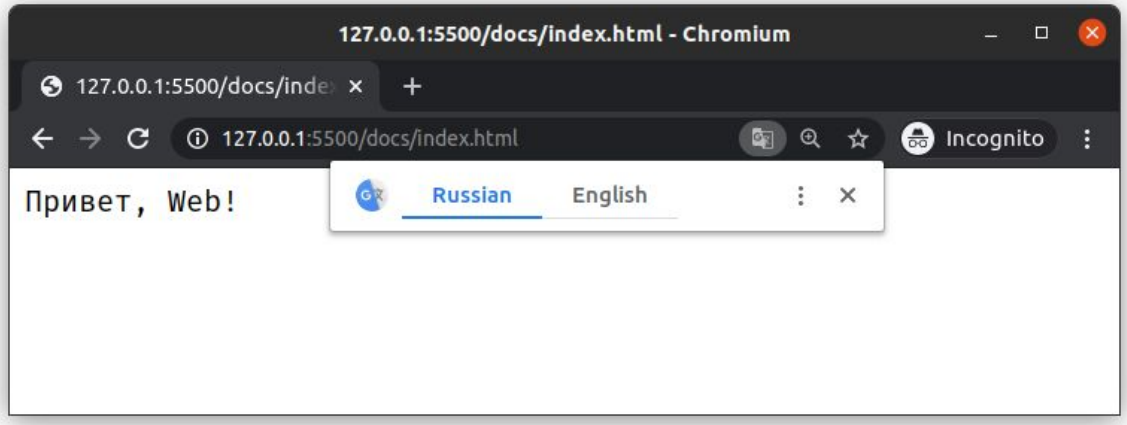
В данном примере html – это родительский элемент для head и body (а они – дочерние по отношению к нему). Это значит, что head и body вложены внутрь html.

html – корневой, это значит, что у него нет родителя (он ни в кого не вложен), кроме того. Напоминаем, что <!DOCTYPE html> – это не элемент.

lang

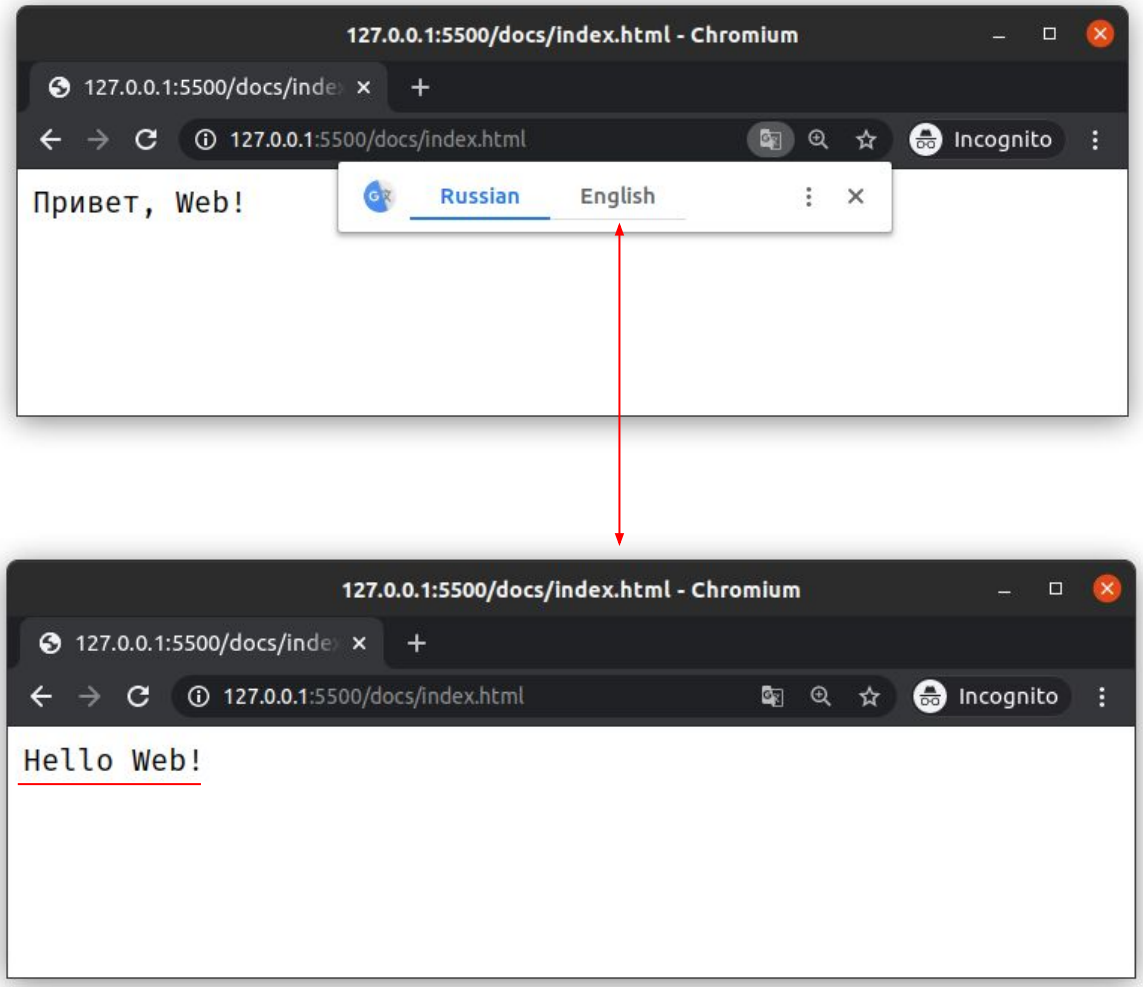
Ключевым атрибутом для элемента html является lang, который определяет, на каком языке написан документ, позволяя элементам вроде переводчиков (встроенных в браузер) или голосовым ридерам предлагать пользователю соответствующие функции перевода или выбирать голоса для озвучки содержимого:

```
<!DOCTYPE html>
<html lang="ru">
<head>
</head>
<body>
  Привет, Web!
</body>
</html>
```



html

```
<!DOCTYPE html>
<html lang="ru">
<head>
</head>
<body>
  Привет, Web!
</body>
</html>
```



lang

Q: как узнать нужные значения?

A: для этого существует отдельный стандарт BCP47, за который отвечает уже известная нам организация IETF.

Основная идея достаточно простая – значение состоит из трёх полей, два из которых необязательны:

код (языковой подтег) - система записи (подтег скрипта) - диалект (подтег региона)

Например:

ru-Cyrl-BY - русский язык, кириллица, Беларусь

lang

В большинстве своём используют только обязательную часть, например:

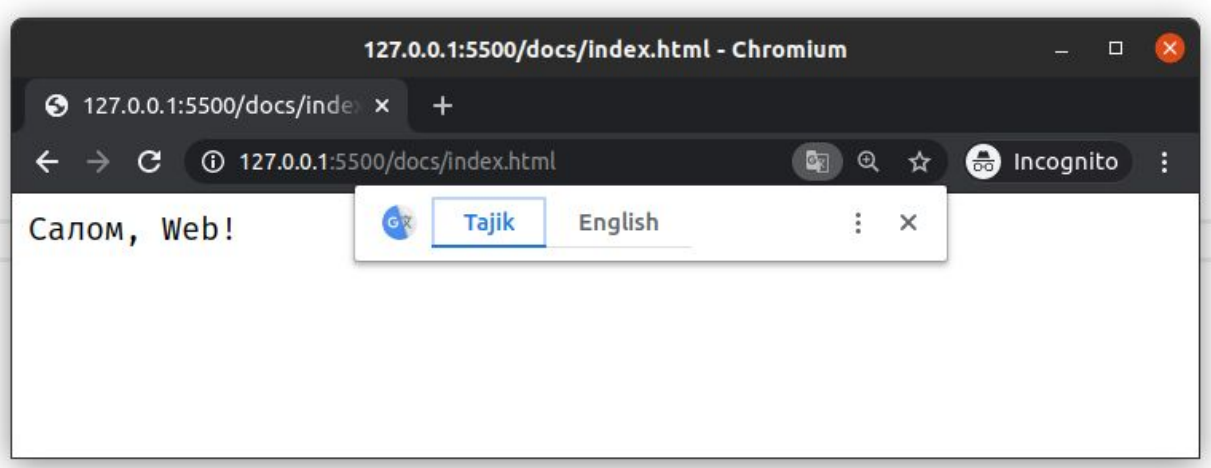
- **en** - английский
- **ru** - русский
- **tg** - таджикский

Q: почему это важно?

A: потому, что если вы неправильно его установите, у пользователя не будет возможности корректно перевести содержимое (а иногда будет всплывать сообщение с предложением перевести содержимое не с того языка, например, с русского на русский, если вы написали lang="en")

lang

```
<!DOCTYPE html>
<html lang="tg">
<head>
</head>
<body>
  Салом, Web!
</body>
</html>
```



html

С html мы разобрались, теперь давайте разберём, что можно писать внутри html.

Для этого в спецификации для каждого тега в разделе описания есть секция content model:

Content model:

A head element followed by a body element.

Если будем переводить, то это значит, что внутри html может быть только элемент head, за которым следует body.

Нарушаем правила

Q: что будет, если мы сделаем по-другому?

A: давайте попробуем. Напишем следующий код:

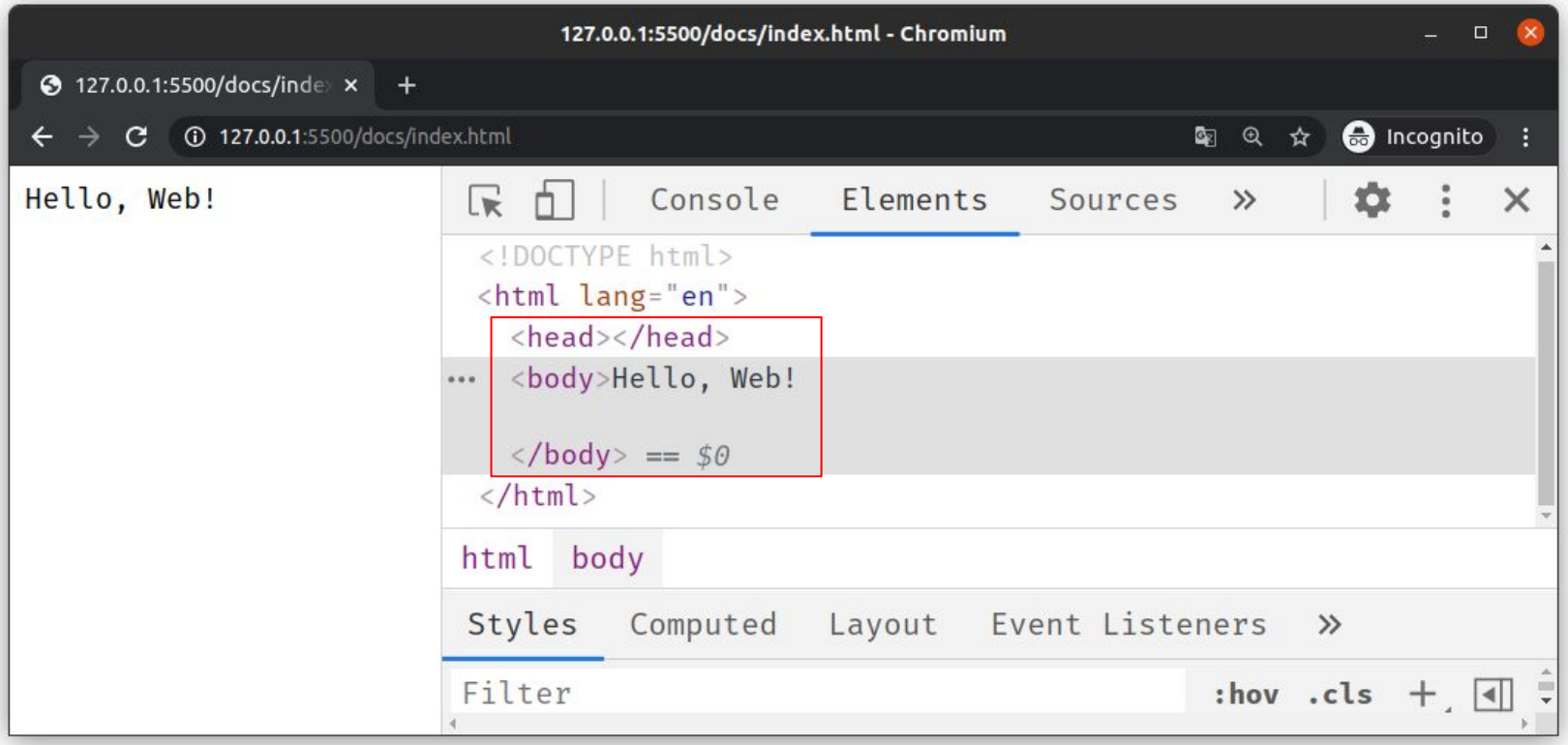
```
<!DOCTYPE html>
<html lang="en">
  Hello, Web!
</html>
```

После чего нажмём сочетание клавиш Ctrl + Shift + I (или F12) в браузере (конечно же у вас должна быть открыта эта страница).

Если у вас ноутбук, то возможно, при нажатии F12 нужно будет ещё нажать клавишу Fn.

Нарушаем правила

Перейдём на вкладку Elements (элементы) и увидим следующее:



Нарушаем правила

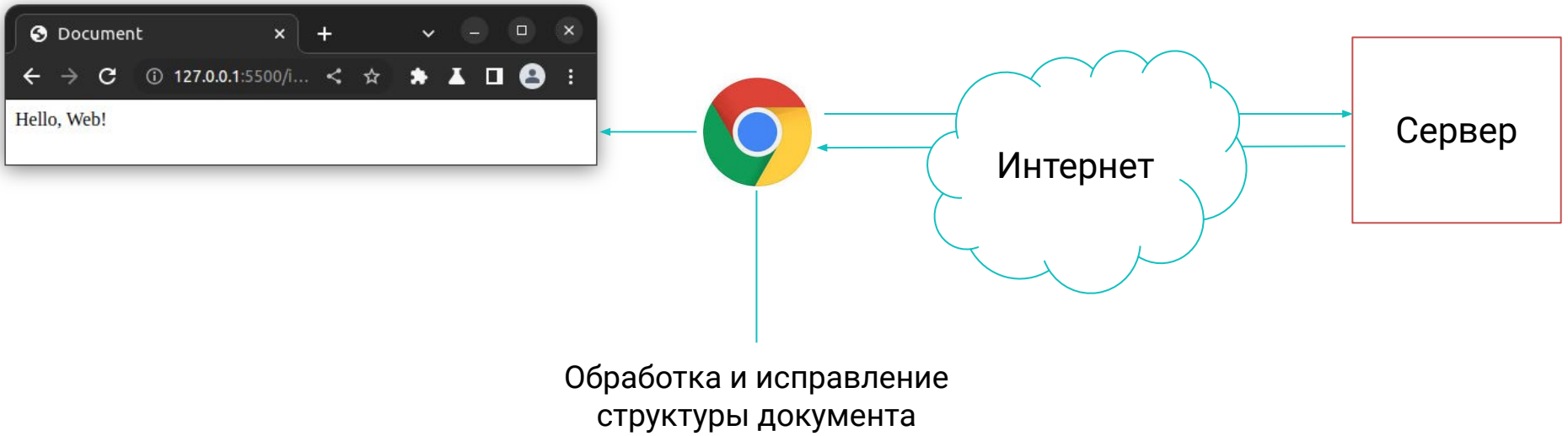
Q: что произошло?

A: браузер "исправил" за нас страницу. А если быть точнее, то (по шагам):

1. Браузер получил страницу и обработал её
2. Понял, что она не соответствует спецификации, поэтому добавил нужные элементы там, где посчитал это правильным
3. Получившуюся страницу нарисовал на экране

Нарушаем правила

Обратите внимание: браузер не может изменить документ, который мы создали в VS Code, он меняет только то, что получил от сервера.



Нарушаем правила

Это одна из ключевых вещей для нас, как для веб-разработчиков: если мы пишем неправильный код, то браузер исправляет его так, как считает нужным.

Поэтому мы стараемся писать код сразу правильно (а как проверять, правильно или нет, мы поговорим чуть позже), чтобы браузер ничего не исправлял.

Элемент head

head

Элемент head отвечает за мета-данные, т.е. данные предназначенные для описания документа, в отличие от body – данных, предназначенных для отображения документа.

title

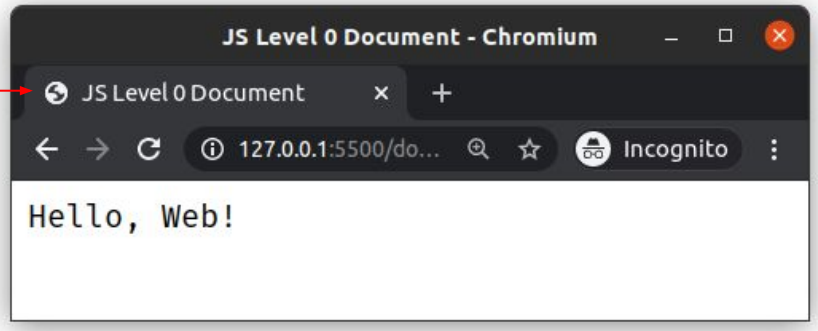
Элемент title является дочерним элементом для head и описывает "название" страницы, которое будет отображаться:

1. На вкладке браузера
2. В истории страниц
3. В избранном
4. И т.д. (например, при передачи ссылку на страницу посредством мессенджеров)*

Примечание*: рассмотрим, когда опубликуем наши документы онлайн.

title

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <title>JS Level 0 Document</title>
</head>
<body>
  Hello, Web!
</body>
</html>
```



Emmet

На прошлых лекциях мы уже говорили об инструменте, который называется Emmet. Он позволял нам с помощью ! + Tab создавать целую структуру страницы.

Давайте поплотнее с ним познакомимся:

1. element + Tab разворачивает нам тег элемента, например: title + Tab будет <title></title> (при этом Emmet уже знает правила, например, для img он не будет создавать закрывающий тег, а для audio и video будет)
2. Если внутри есть контент (содержимое в виде текста), то можно набрать element{контент} + Tab, например title{Document Title} + Tab будет <title>Document Title</title>

Emmet

Важно: обязательно с первых дней привыкайте пользоваться Emmet'ом. Нужно себя "заставлять" писать сразу именно сокращениями Emmet – первое время это будет дольше, чем написать "всё руками", но через несколько недель практики ваша скорость будет значительно выше (это примерно как слепая десятипальцевая печать против обычной печати двумя пальцами).

Напоминаем, что шпаргалка по его использованию находится по ссылке. Если документ по ссылке не открывается, используйте сохранённую нами копию: https://alif-skills.pro/media/emmet.pdf

meta

Элемент meta достаточно сложный, поскольку может использоваться сразу для нескольких целей. Вот ключевые:

1. Задание кодировки документа
2. Задание параметров отображения

Кодировка

Для начала сделаем небольшой экскурс в хранение информации на компьютере и разберём такое понятие, как кодировка (мы будем часто с ним сталкиваться).

Хранение информации

Внутри компьютера информация хранится в виде бит – бит это самая наименьшая единица хранения информации. Бит может быть либо установлен (хранить значение 1) или не установлен (хранить значение 0). Других значений бит хранить не может, только 0 или 1.

Вспомните, как считают дети на пальцах – они загибают пальчик, когда прибавляют единичку (считают). Вот бит – это и есть этот самый пальчик: если пальчик загнут (говорят, что в бите хранится число 1) – то единичка установлена, если не загнут, то не установлена (т.е. в бите хранится число 0).

Хранение информации

Именно таким образом всё устроено. И если мы снова посмотрим, как считают дети, то увидим, что у них 10 пальчиков и они по порядку загибают их:

не загнут	не загнут	не загнут	не загнут	не загнут	не загнут	не загнут	не загнут	загнут	загнут
0	0	0	0	0	0	0	0	1	1

Итого получается 1 + 1 = 2.

Обратите внимание, если мы просто запишем это как число (выпишем нолики и единицы), то получится:

0000000011

Хранение информации

С помощью одного пальчика можно хранить всего две цифры – 0 (не загнут) и 1 (загнут). Такую систему, в которой на одной позиции может быть всего одна из двух цифр - называют двоичной (т.к. цифры всего две).

Мы же с вами используем десятичную, например, число 23. В каждой позиции у нас может быть одна из следующих цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (всего 10). Поэтому такую систему счисления называют десятичной (т.к. цифр уже 10).

Обычно, когда пишут числа не в десятичной системе счисления, в программировании сразу записывают, в какой системе счисления записано число: 11_2 – значит, что число записано в двоичной системе счисления.

Хранение информации

А теперь давайте посмотрим, как "считаются" наши обычные числа в десятичной системе счисления: возьмём для примера число 23.

Напишем над его цифрами сверху справа-налево позиции, начиная с нуля:

позиция	1	0
цифры	2	3

И на самом деле, всё считается как:

2 * 10 ** 1 + 3 * 10 ** 0, где ** – это оператор возведения в степень (см. далее).

Давайте разбираться детальнее.

Хранение информации

2 * 10 ** 1 + 3 * 10 ** 0.

Q: почему мы умножаем на 10?

A: потому, что десятичная система счисления (10 цифр) и "вес" каждой цифры зависит от того, на какой она позиции, например в числе 9001 "вес" 9 гораздо больше, чем 1.

Это легко понять (идею про вес позиции), если представить что это число – количество денег, например, в сомони. И если вы 9 замените на 0, то у вас останется 0001 (т.е. 1 сомони, по сравнению с тем, что было), а вот если вы 1 замените на 0, то не так уж и страшно – 9000. Поэтому когда вам предложат "занулить" какую-то позицию, вы наверняка выберите наименьшую.

Хранение информации

2 * 10 ** 1 + 3 * 10 ** 0.

Q: что за операция **?

A: Это операция возведения в степень (степень – это то, что стоит после **).

Давайте запомним простые правила:

1. Любое неотрицательное целое число в нулевой степени будет равно 1 (например, 10 ** 0 будет 1)
2. Любое неотрицательное целое число в степени 1 будет равно самому себе (например, 10 ** 1 будет 10)
3. Любое неотрицательное число в степени больше единица нужно умножить само на себя столько раз, сколько написано после ** (например, 10 ** 2 это 10 * 10 – будет 100)

Хранение данных

Q: но как это соотносится с тем, что мы обсуждали про биты?

A: дело в том, что те же биты считаются так же, как мы только что обсуждали десятичные, так же работает и с бинарными данными, единственное, всё умножается на 2:

10_2 = 1 * 2 ** 1 + 0 * 2 ** 0 = 2_{10}

Таким образом, компьютер всё хранит в битах, но для нас, когда мы смотрим на результат, всё переводится в десятичную систему счисления.

Байт

Поскольку считать биты неудобно, придумали более крупную единицу – байт. Байт – это не одна позиция (как бит), а целых 8.

Поэтому всё измеряют в байтах (или килобайтах – 1024 байт, мегабайтах – 1024 килобайта, гигабайтах – 1024 мегабайта, террабайтах – 1024 гигабайта и т.д.).

Буквы

С помощью такого представления можно записывать числа, но нам не всегда достаточно чисел. Хотелось бы возможности хранить и строки, и цвета. Как это сделать?

Хранение информации

Всё достаточно просто – можно договориться, что мы придумаем табличку, в которой напишем, какому символу какой набор 0 и 1 соответствует:

Символ	Код
A	01000001
B	01000010
C	01000011
...	...

Буквы

Таким образом получается, что на хранение одного символа (или буквы) нам нужно 1 байт (т.е. 8 бит).

Сколько тогда буквы мы сможем закодировать?

Ответ достаточно простой – надо сосчитать все возможные значения от 00000000 до 11111111. Конечно же, вручную это делать неудобно, поэтому мы сразу скажем вам ответ – 256. Всего 256 (в то время как надо суметь закодировать символы из разных языков мира)!

Буквы

Q: а если нам нужно больше, чем 256 букв? Например, для арабского и китайского языков.

A: ответ достаточно простой – для таких символов и букв мы можем использовать не один байт, а несколько. Главное договориться, чтобы программа, которая будет читать байты и отображать нам в виде текста, понимала, какую табличку мы используем для перевода байт в буквы.

ASCII

Изначально существовало достаточно много кодировок. В Америке была распространена кодировка ASCII (American standard code for information interchange), где 128 символов использовались для латинских символов, цифр и знаков.

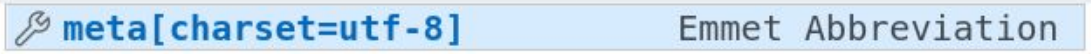
Но, как вы понимаете, это не подходит для современного мира, в котором гораздо больше символов (+ есть Emoji и т.д.).

UTF-8

Самая распространённая кодировка на данный момент – UTF-8, которая на каждый символ (в зависимости от символа) тратит от 1 до 4 байт (причём первые 128 символов соответствуют ASCII).

Именно она чаще всего и указывается в атрибуте charset элемента meta:

```
meta[charset=utf-8]
```



+ Tab

```
<meta charset="utf-8">
```

Обратите внимание, что в самом VS Code внизу должна быть выставлена такая же кодировка:

Internet Explorer

В некоторых случаях, Emmet вам будет вставлять следующий элемент:

```
<meta http-equiv="X-UA-Compatible" content="IE=edge">
```

Это специфичный для устаревшего браузера Internet Explorer (сокращённо IE), элемент, сообщающий этому браузеру, что нужно использовать самый современный вариант отображения.

Его можно удалить из документа.

meta name="viewport"

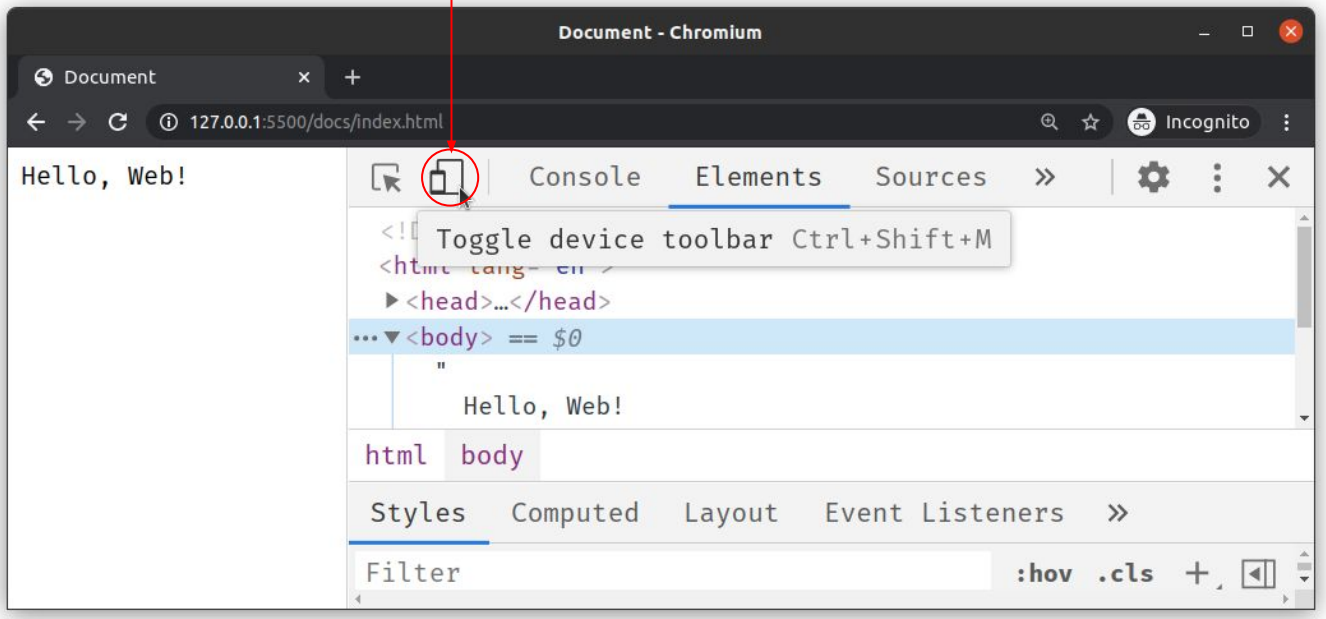
А теперь окунёмся в достаточно интересную тему: вопросы отображения нашей страницы в различных устройствах.

Наверняка, для вас не секрет, что современные веб-страницы можно просматривать с:

1. Компьютеров
2. Планшетов
3. Смартфонов
4. И других устройств (например, телевизоров)

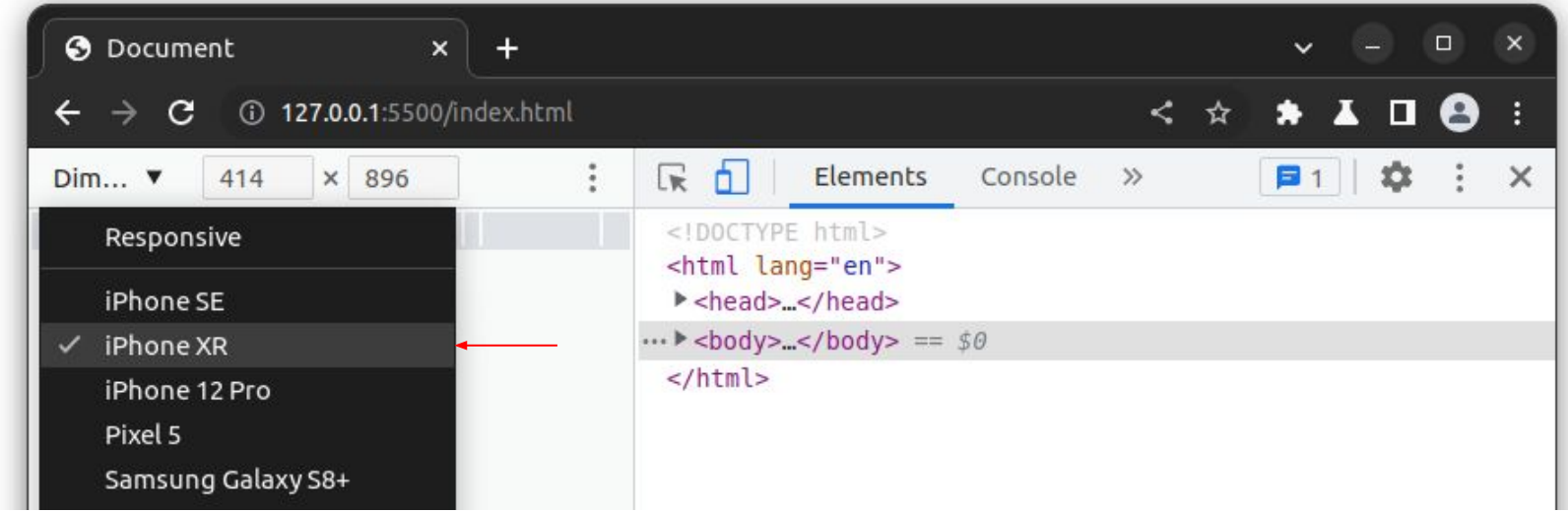
meta name="viewport"

В браузер встроены "эмуляторы", которые позволяют посмотреть, как будет выглядеть наша страница на других устройствах – для этого надо нажать Ctrl + Shift + I (или F12) и нажать на иконку с планшетом или смартфоном (или Ctrl + Shift + M):

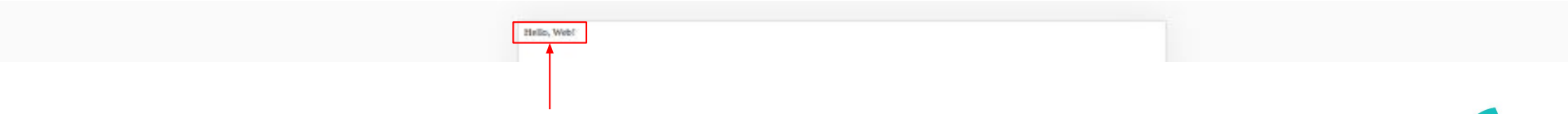


meta name="viewport"

Выберите из выпадающего списка какое-нибудь устройство, например, iPhone XR:



Вы увидите, что при содержимом элемента head как на скриншоте, текст почти не читаем:

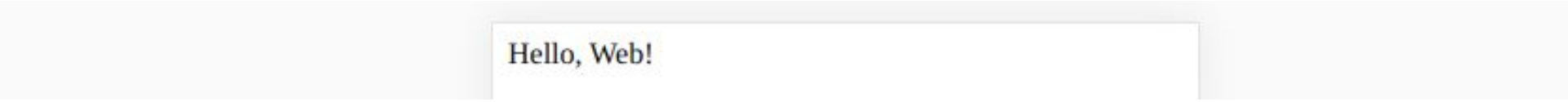


meta name="viewport"

Но если мы добавим следующий элемент:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

То всё станет гораздо лучше:



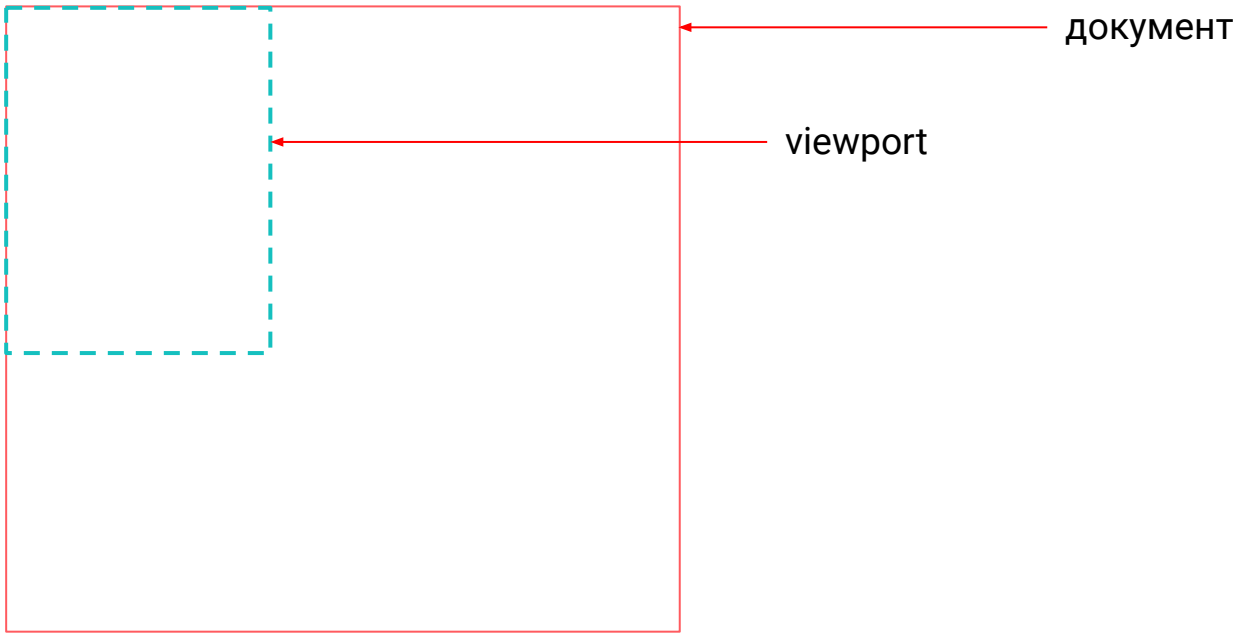
Теперь давайте разберёмся с тем, что в нём написано:

- viewport – означает область просмотра (см.следующий слайд)
- with=device-width – означает, что ширина должна быть равна ширине устройства
- initial-scale=1.0 – означает, что начальное масштабирование должно быть 1.0 (не масштабируем – т.е. не увеличиваем и не уменьшаем), на смартфонах вы можете масштабировать страницу с помощью соответствующего жеста

viewport

Q: что такое viewport?

A: это область, через которую мы смотрим на страницу, пояснить это можно с помощью следующей картинки (пример, когда ширина viewport и документа различаются):



head

Таким образом, мы с вами обсудили базовое содержимое элемента head, которое генерируется Emmet.

body

body

Теперь пришла очередь элемента body и тех элементов, которые могут входить в него.

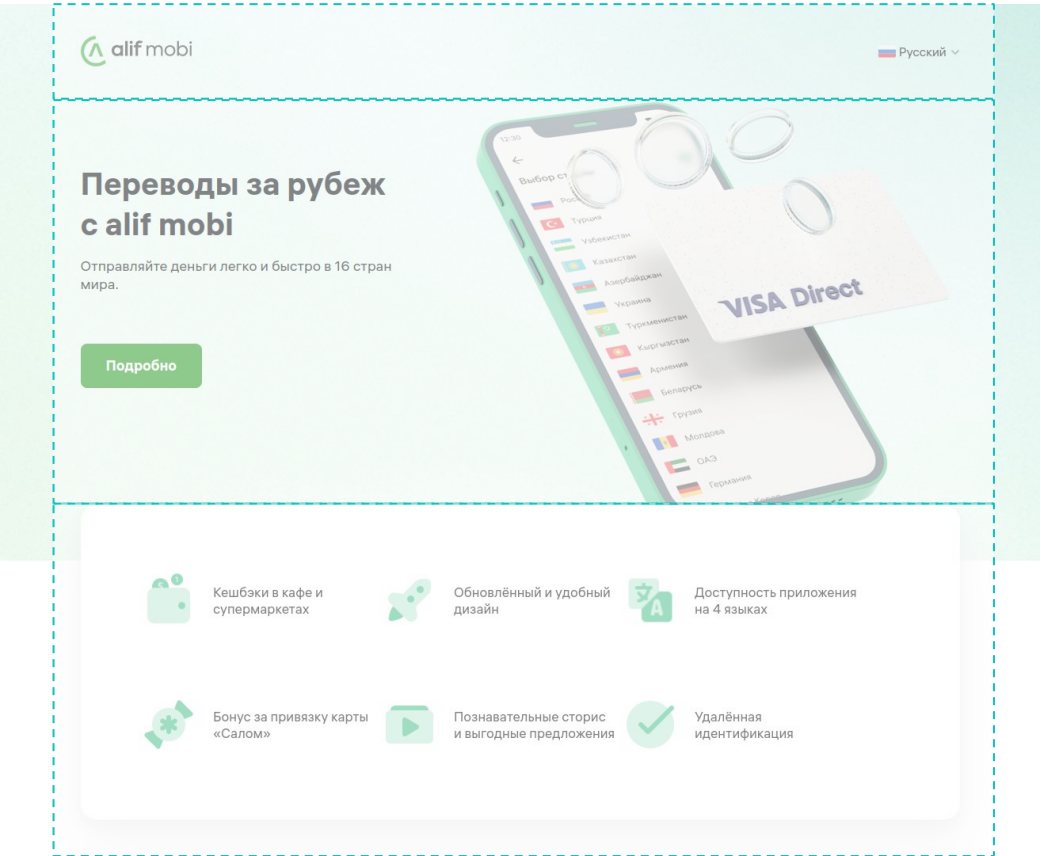
body – это контейнер (т.е. элемент, в который можно вкладывать другие элементы) содержимого страницы, которое (содержимое) будет отображено веб-браузером.

Т.е. в простейшем случае - всё, что мы напишем внутри body, должно отображаться на странице*.

Примечание*: конечно же, есть исключения.

div vs semantic

Помимо специализированных элементов, предназначенных для отображения конкретных элементов (например, ссылки, изображения, аудио и видео), нам нужно формировать структуру страницы, разделяя её на условные блоки:



div vs semantic

И дальше весь мир людей, создающих HTML документы, делится на две части:

1. Тех, кто использует div
2. Тех, кто использует семантические элементы (т.е. имеющие определённый смысл)

div

div – это контейнер без какого-то специального предназначения. Т.е. внутрь мы можем поместить что угодно: другие элементы, текст и т.д. В итоге весь наш документ превращается в набор div'ов + специализированные элементы (вроде img, video, audio, которые div'ом заменить не получится).

В спецификации по этому поводу сказано следующее:

Note

> *Authors are strongly encouraged to view the `div` element as an element of last resort, for when no other element is suitable. Use of more appropriate elements instead of the `div` element leads to better accessibility for readers and easier maintainability for authors.*

В переводе: используйте элемент div только тогда, когда ни один из других элементов не подходит.

"шапка" с меню

основное содержимое

подвал

Я желаю: Делать покупки Зарабатывать Заботиться

Одна карта — три валюты

Visa Алифа для всего — переводов, поездок, покупок в интернете и в рассрочку

Узнать больше >

Карта «Салом»

Покупайте товары и услуги в рассрочку быстро, без предоплаты до 24 месяцев

Узнать больше >

Онлайн-покупки в рассрочку

Заказывайте бытовую технику, смартфоны и другие товары в рассрочку, не выходя из дома

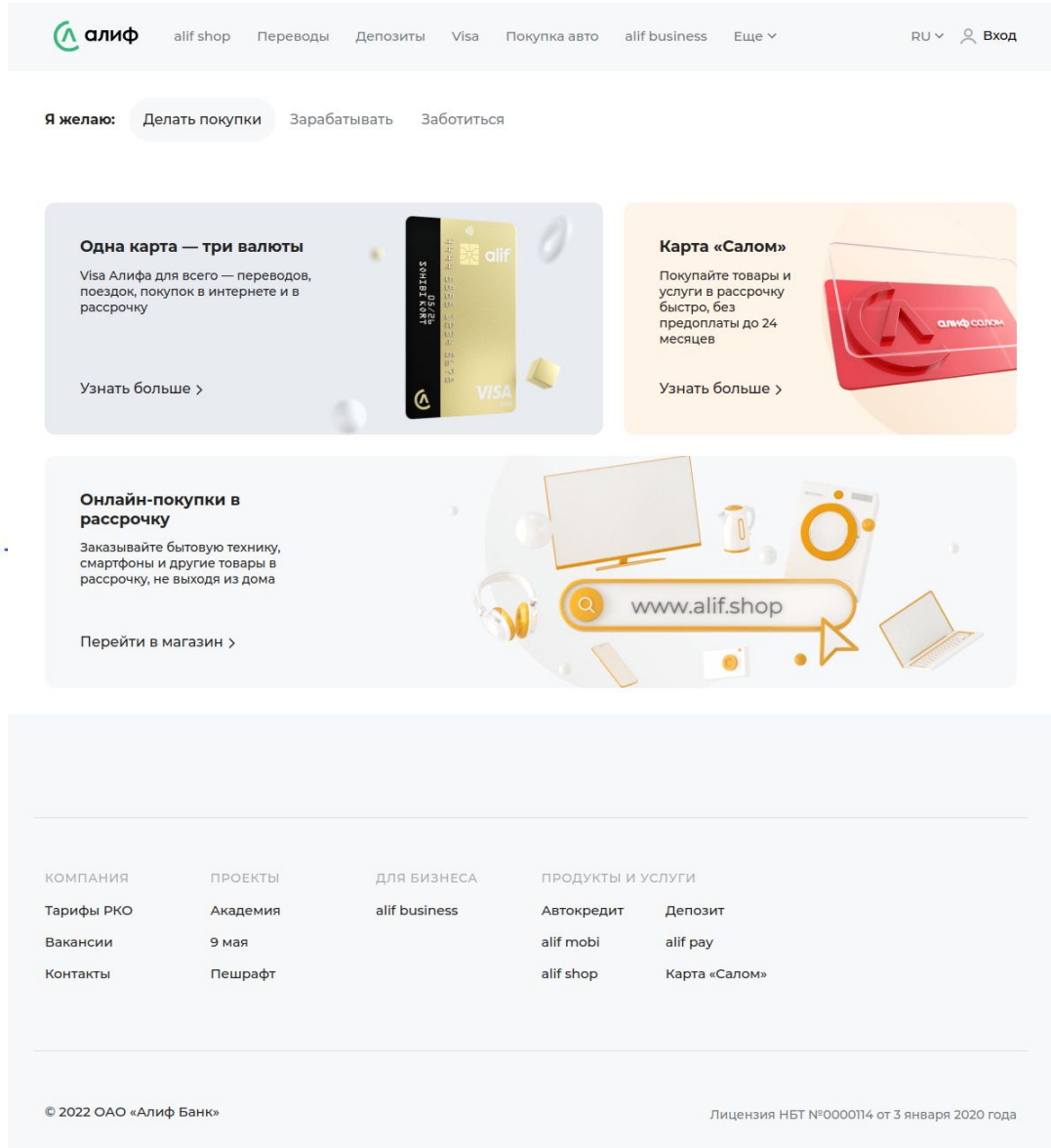
Перейти в магазин >

Лицензия НБТ №0000114 от 3 января 2020 года

div

В реальности вы достаточно часто будете видеть, что весь документ собран на div'ax, например:

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device·
  <title>Web страница на div'ax</title>
</head>
<body>
  <div>Шапка с меню</div>
  <div>Основное содержимое</div>
  <div>Подвал</div>
</body>
</html>
```



div

Важно: div'ы – это не хорошо и не плохо. Это выбранный стиль, который позволяет нам создавать документы очень быстро: нам не нужно долго думать о том, какой элемент использовать:

1. Мы берём специализированный (например, гиперссылку, форму, видео, аудио или картинку), если нужна спец.функциональность
2. Во всех остальных случаях берём div

semantic

Альтернативный подход – использование семантических элементов, которые позволяют указать на предназначение их содержимого. Например:

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Web страница на div'ax</title>
</head>
<body>
  <header>Шапка с меню</header>
  <main>Основное содержимое</main>
  <footer>Подвал</footer>
</body>
</html>
```

При таком подходе мы буквально "размечаем" смысловые блоки нашей страницы для браузеров и, например, поисковых систем.

semantic

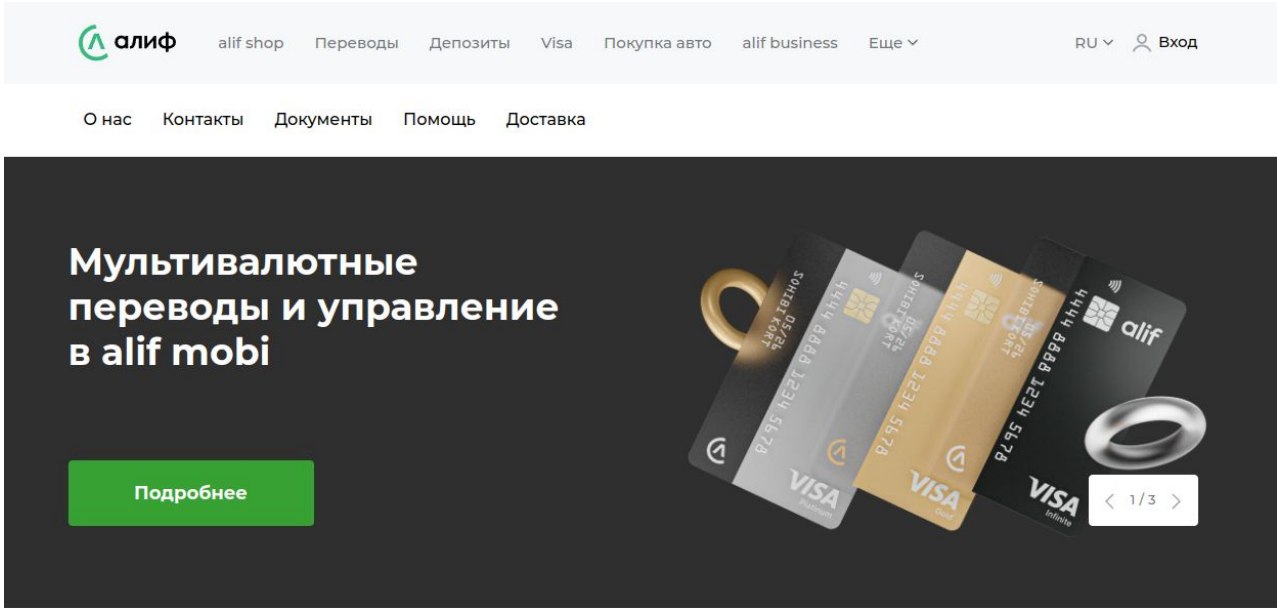
Важно: чтобы не усложнять, мы пока будем рассматривать вариант простых документов, когда все описываемые далее нами семантические элементы находятся непосредственно внутри body (т.е. у них нет других родителей, кроме body).

header

Если header располагается прямо внутри body (есть и другие возможности), то это вводная часть страницы, которая может содержать:

1. Логотип
2. Навигационные ссылки (т.е. меню)
3. Баннеры

Чаще всего это то, что целиком (или частично) повторяется на нескольких веб-страницах:



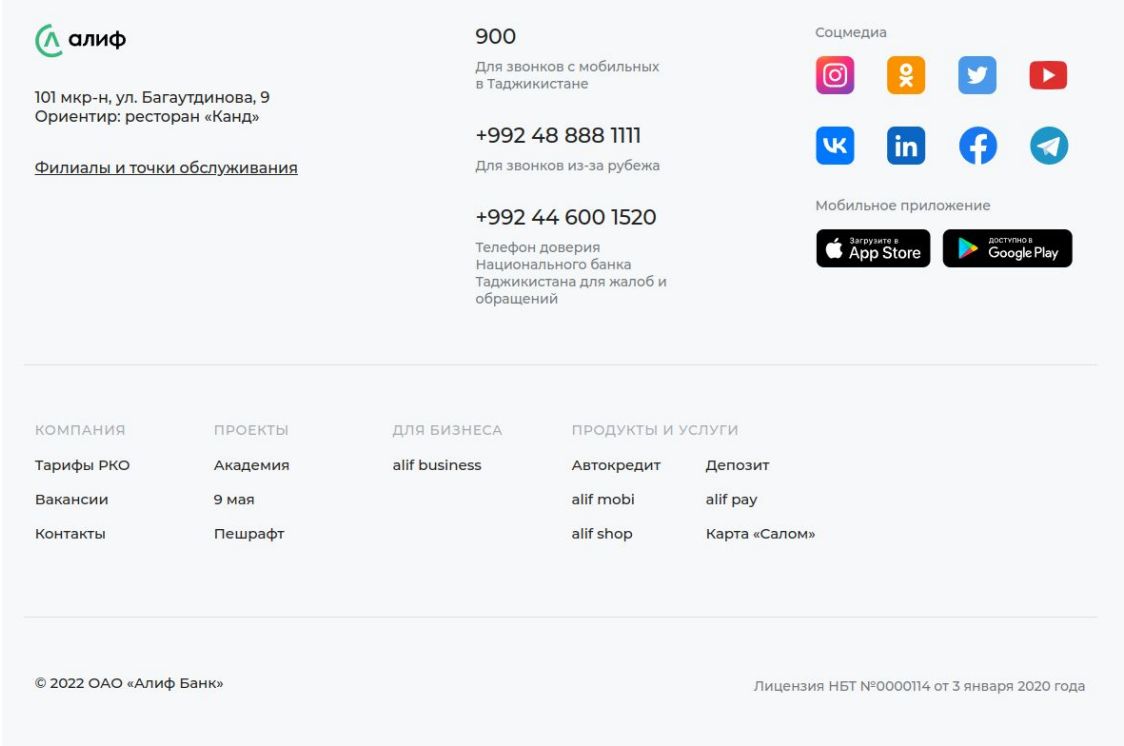
footer

Если footer располагается прямо внутри body (есть и другие возможности), то это нижняя часть страницы, которая может содержать:

1. Логотип
2. Навигационные ссылки (т.е. меню)
3. Контакты и т.д.

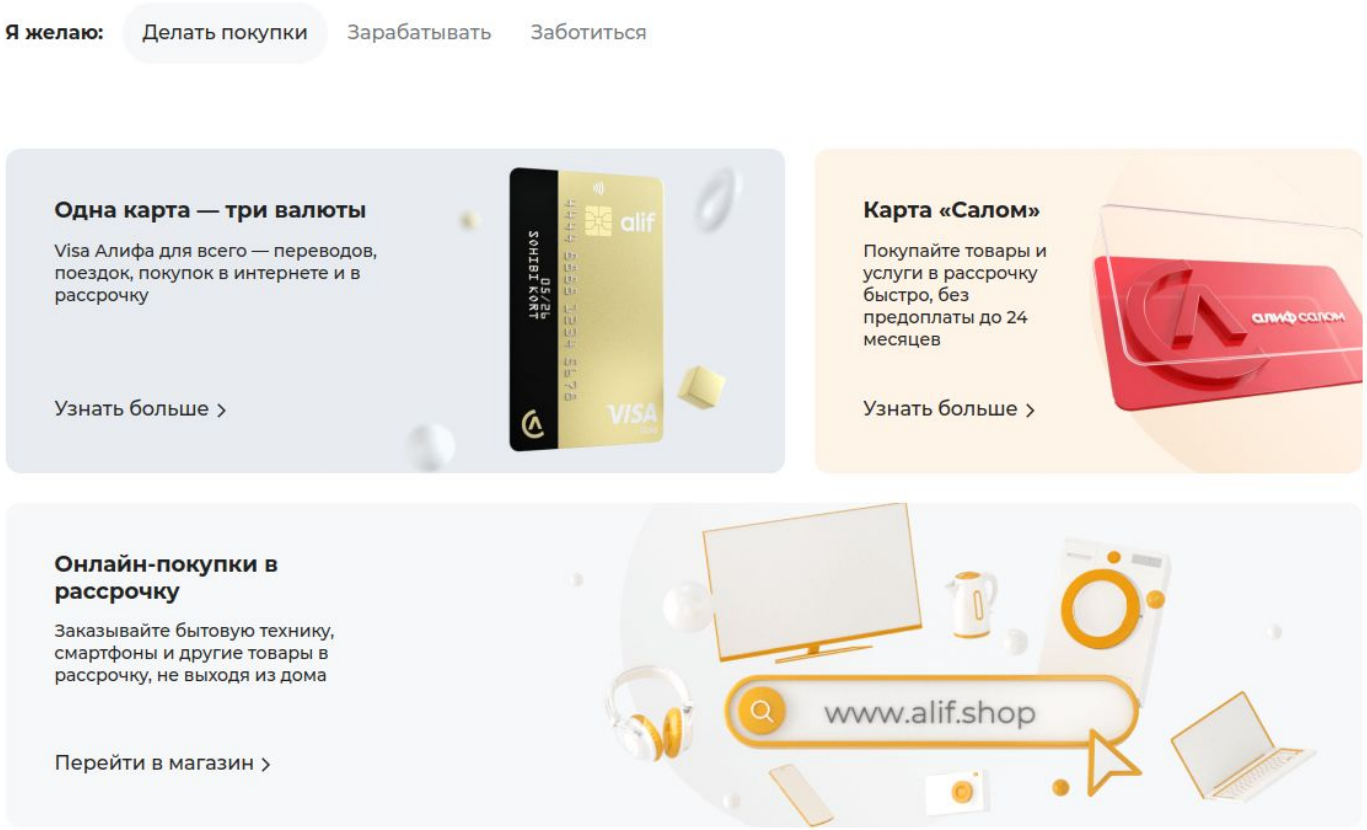
Чаще всего это то, что целиком (или частично) повторяется на нескольких веб-страницах:

В этом примере footer может быть как с блоком контактов, так и без него.



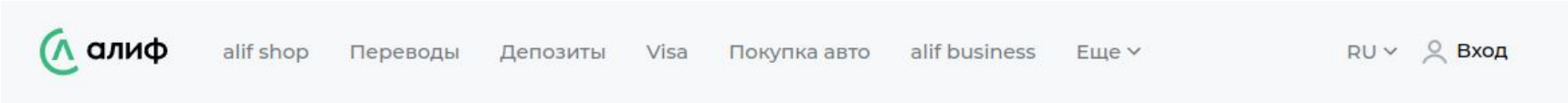
main

Если main располагается прямо внутри body (есть и другие возможности), то это основное и уникальное содержимое страницы:

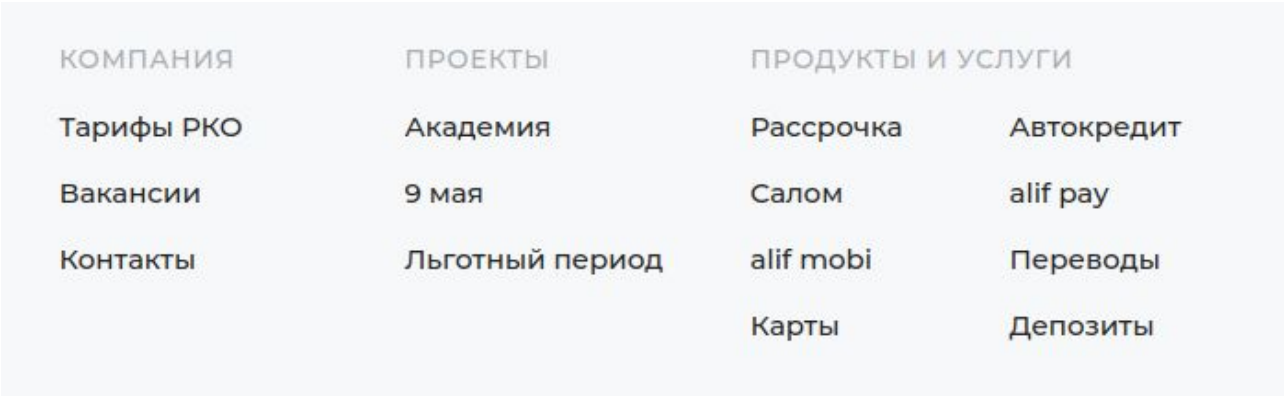


nav

Элемент nav отвечает за выделение основных блоков навигации (позволяющих перемещаться по веб-сайту):



Здесь ключевое слово - основных. То, что находится в footer обычно не является основной навигацией (чаще всего - это дублирующая):



semantic

В современном мире подход с семантической разметкой документов считается более предпочтительным. Но он так же требует от вас времени (вам приходится задумываться, где у вас начинается и заканчивается header, footer и main).

Нужно отметить, что ключевая проблема – нет чётких правил вроде: вот здесь обязательно должен быть такой семантический элемент. Поэтому придётся потратить время на подбор нужного элемента. От того, что мы напишем другой – ничего не сломается.

ИТОГИ

Итоги

В этой лекции мы детально разобрали базовую структуру страницах и поговорили о двух подходах к разметке страницы:

- div
- semantic elements

ДОМАШНЕЕ ЗАДАНИЕ

Орг.моменты

Практикум состоит из 8 обязательных занятий. Мы выкладываем новые занятия каждую субботу в 12:00 (по Душанбе).

Каждую пятницу в 23:59 (по Душанбе) дедлайн сдачи домашнего задания. Дедлайн – это предельный срок, до которого вы должны сдать ДЗ.

Если не успеете сдать в срок домашнее задания, тогда этот практикум будет для вас закончен и вы сможете зарегистрироваться на запуск следующего через несколько месяцев.

Все вопросы вы сможете задавать в Телеграм канале.

ДЗ: Язык и кодировка

Создайте проект аналогично тому, как мы это делали на лекции. Установите у нужных элементов правильные атрибуты, чтобы получилось:

1. Язык страницы – таджикский
2. Кодировка страницы – utf-8

У вас должно быть два элемента meta (как в примерах из лекции – для кодировки и для viewport), если VS Code генерирует дополнительные – удалите их.

Содержимое тега body должно быть равно: Салом, Web!

Содержимое тега title должно быть: Alif Skills

ДЗ: semantic

Создайте проект аналогично тому, как мы это делали на лекции. Внутри элемента body разместите подходящие по смыслу элементы (не div), которые позволили бы создать следующие три раздела:

1. "Шапка" с логотипом (картинка): 

2. Основное содержимое с текстом: Alif Skills – это онлайн-платформа для получения практических навыков в сфере IT. Мы делимся своим опытом и знаниями, чтобы помочь вам стать профессиональными разработчиками.
3. "Подвал" сайта с текстом: © 2022 ОО «Алиф». (точка должна быть, кавычки "ёлочкой" нужно будет найти самостоятельно).

Для отображения логотипа используйте следующий адрес: https://alif-skills.pro/media/alif-skills.svg. Атрибут alt должен быть равен: Alif Skills.

ДЗ: semantic

Для ввода спец.символов (вроде тире, знаков © и т.д.) существуют специальные сущности. Их список и значение вы можете найти тут.

В HTML-разметке можно будет писать ©, а на странице будет отображаться ©.

ДЗ: div

Создайте проект аналогично тому, как мы это делали на лекции. Внутри элемента body разместите элементы div, которые позволили бы создать следующие три раздела:

1. "Шапка" с логотипом (картинка): 

2. Основное содержимое с текстом: Alif Skills – это онлайн-платформа для получения практических навыков в сфере IT. Мы делимся своим опытом и знаниями, чтобы помочь вам стать профессиональными разработчиками.
3. "Подвал" сайта с текстом: © 2022 ОО «Алиф». (точка должна быть, кавычки "ёлочкой" нужно будет найти самостоятельно)

Для отображения логотипа используйте следующий адрес: https://alif-skills.pro/media/alif-skills.svg. Атрибут alt должен быть равен: Alif Skills.

Спасибо за внимание

alif skills

2022г.